

LAPORAN MINGGU 8

Praktikum Fundamental Programming Structures In Java

Teknik Pemograman

Tugas ini untuk memenuhi Mata Kuliah Teknik Pemograman Praktek



Disusun Oleh :

Reka Briyan Cahya Heryana – 211524024

**PROGRAM STUDI D4 TEKNIK INFOMATIKA
JURUSAN TEKNIK KOMPUTER DAN INFORMATIKA
POLITEKNIK NEGERI BANDUNG**

2022

TASK 1.1 – 1.3

1. Hasil Program

```
run:
Cylinder: radius=1.0 height=1.0 base area=12.566370614359172 volume=12.566370614359172
Cylinder : subclass of Circle[radius=1.0 color=red]height=1.0

Cylinder: radius=1.0 height=10.0 base area=69.11503837897544 volume=691.1503837897544
Cylinder : subclass of Circle[radius=1.0 color=red]height=10.0

Cylinder: radius=2.0 height=10.0 base area=150.79644737231007 volume=1507.9644737231006
Cylinder : subclass of Circle[radius=2.0 color=red]height=10.0

BUILD SUCCESSFUL (total time: 0 seconds)
```

2. Jawaban dari Soal yang dipertanyakan

Circle.java

```
public class Circle { // Save as "Circle.java"
    // private instance variable, not accessible from outside this class
    private double radius;
    private String color;

    public String getColor(){
        return color;
    }

    public void setColor(String color){
        this.color = color;
    }

    // Constructors (overloaded)
    /** Constructs a Circle instance with default value for radius and color */
    public Circle() { // 1st (default) constructor
        radius = 1.0;
        color = "red";
    }

    /** Constructs a Circle instance with the given radius and default color */
    public Circle(double r) { // 2nd Constructor
        radius = r;
        color = "red";
    }
}
```

Cylinder.java

```
public class Cylinder extends Circle { // Save as "Cylinder.java"
    private double height; // private variable

    // Constructor with default color, radius and height
    public Cylinder() {
        super(); // call superclass no-arg constructor Circle()
        height = 1.0;
    }

    // Constructor with default radius, color but gives height
    public Cylinder(double height) {
        super(); // call superclass no-arg constructor Circle()
        this.height = height;
    }

    // Constructor with default color, but given radius, height
    public Cylinder(double radius, double height) {
        super(radius); // call superclass constructor Circle(r)
        this.height = height;
    }

    // A public method for retrieving the height
    public double getHeight() {
        return height;
    }
}
```

TestCylinder.java

```
public class TestCylinder { // save as "TestCylinder.java"
    public static void main (String[] args) {
        // Declare and allocate a new instance of cylinder
        // with default color, radius, and height
        Cylinder c1 = new Cylinder();
        System.out.println("Cylinder:"
            + " radius=" + c1.getRadius()
            + " height=" + c1.getHeight()
            + " base area=" + c1.getArea()
            + " volume=" + c1.getVolume()
            + "\n" + c1.toString() + "\n");

        // Declare and allocate a new instance of cylinder
        // specifying height, with default color and radius
        Cylinder c2 = new Cylinder(10.0);
        System.out.println("cylinder:"
            + " radius=" + c2.getRadius()
            + " height=" + c2.getHeight()
            + " base area=" + c2.getArea()
            + " volume=" + c2.getVolume()
            + "\n" + c2.toString() + "\n");

        // Declare and allocate a new instance of cylinder
        // specifying radius and height, with default color
        Cylinder c3 = new Cylinder(2.0, 10.0);
        System.out.println("Cylinder:"
```

3. Masalah yang dihadapi

- Task 1.1
Saya memberikan variabel color bertipe string, lalu memberikan 3 konstruktor pada kelas circle serta saya menambahkan getter dan setter.
- Task 1.2
Saya menggunakan override di kelas cylinder dan TestCylinder (Main).
Overriding ini merupakan sebuah fitur yang dimana sebuah subkelas atau anak kelas yang menyediakan sebuah implementasi yang jelas dari metode yang sudah tersedia oleh salah satu dari super kelas atau parent kelas.
- Task 1.3
Menambahkan metode toString pada kelas TestCylinder (Main). Method ini digunakan untuk merepresentasikan sebuah objek, kedalam tipe String. Jika kita ingin menampilkan sebuah objek secara implisit dan sebenarnya compiler sudah memanggil method toString(). Untuk memunculkan ke layar gunakan toString().

4. Solusi dari Masalah yang Dihadapi

- Task 1.1

```
public String getColor(){
    return color;
}

public void setColor(String color){
    this.color = color;
}
```

- Task 1.2

```
@Override
public double getArea() {
    return (2*Math.PI*getRadius())+(2*super.getArea());
}

@Override
public String toString() {
    return "Cylinder: subclass of " + super.toString() + "Height=" + height;
}
```

- Task 1.3

```
+ "\n" + c1.toString() + "\n";
+ "\n" + c2.toString() + "\n";
+ "\n" + c3.toString() + "\n";
```

5. Teman yang Membantu

- Tidak ada

Task 2.1

1. Hasil Program

```
INFO:
A Shape with color of green and filled
A Shape with color of red and not filled
A Circle with radius = 7.0, which is a subclass of A Shape with color of red and not filled
Area = 153.93804002589888
Perilling = 42.882297150257104

A Rectangle with width = 3.0 and length = 4.0, which is a subclass of A Shape with color of blue and filled
Area = 12.0
Perilling = 14.0

A Square with side = 3.0, which is a subclass of A Rectangle with width = 3.0 and length = 3.0, which is a subclass of A Shape with color of yellow and filled
Area = 9.0
Perilling = 12.0
INFO: SUCCESSFUL (total time: 0 seconds)
```

2. Jawaban dari Soal yang Dipertanyakan

Circle.java

```
1  package task2;
2
3  public class Circle extends Shape {
4      // Instance Variables
5      private double radius;
6
7      // Constructors
8      public Circle() { // 1st constructor
9          radius = 1.0;
10     }
11
12     public Circle(double radius) { // 2nd constructor
13         this.radius = radius;
14     }
15
16     public Circle(double radius, String color, boolean filled) { // 3rd constructor
17         super(color, filled);
18         this.radius = radius;
19     }
20
21     // Define and Getter
22     public double getRadius() {
23         return radius;
24     }
25
26     public void setRadius(double radius) {
27         this.radius = radius;
28     }
29
30     /** Return the area of this Circle instance */
31     public double getArea() {
32         return radius*radius*Math.PI;
33     }
34
35     /** Return the perimeter of this Circle instance */
36     public double getPerimeter() {
37         return 2*Math.PI*radius;
38     }
39
40     /** Return a self-descriptive string of this instance in the form of
41     Circle[Shape{color=*, filled=*, radius=**}] */
42     public String toString() {
43         return "A Circle with radius = " + getRadius() + ", which is a subclass"
44             + " of " + super.toString();
45     }
46 }
47
```

Shape.java

```
1 public class Shape {
2     // Instance Variables
3     private String color;
4     private boolean filled;
5
6     // Constructors
7     public Shape() { // 1st constructor
8         color = "green";
9         filled = true;
10    }
11
12    public Shape(String color, boolean filled) { // 2nd constructor
13        this.color = color;
14        this.filled = filled;
15    }
16
17    // Getter and Setter
18    public String getColor() {
19        return color;
20    }
21
22    public void setColor(String color) {
23        this.color = color;
24    }
25
26    public boolean isFilled() {
27        return filled;
28    }
29
30    public void setFilled(boolean filled) {
31        this.filled = filled;
32    }
33
34    /** Return a self-descriptive string of this instance in the form of
35     * Circle(radius=r,color=c) */
36    public String toString() {
37        if(isFilled()) {
38            return "A Shape with color of " + getColor() + " and filled";
39        }
40        else {
41            return "A Shape with color of " + getColor() + " and not filled";
42        }
43    }
44 }
```

Rectangle.java

```
1 public class Rectangle extends Shape {
2     // Instance Variables
3     private double width;
4     private double length;
5
6     // Constructors
7     public Rectangle() {
8         width = 1.0;
9         length = 1.0;
10    }
11
12    public Rectangle(double width, double length) {
13        this.width = width;
14        this.length = length;
15    }
16
17    public Rectangle(double width, double length, String color, boolean filled) {
18        super(color, filled);
19        this.width = width;
20        this.length = length;
21    }
22
23    // Getter and Setter
24    public double getWidth() {
25        return width;
26    }
27 }
```

```

36     public void setWidth(double width) {
37         this.width = width;
38     }
39
40     public double getLength() {
41         return length;
42     }
43
44     public void setLength(double length) {
45         this.length = length;
46     }
47
48     /** Returns the area of this Circle instance */
49     public double getArea() {
50         return width*length;
51     }
52
53     /** Returns the perimeter of this Circle instance */
54     public double getPerimeter() {
55         return (2*width) + (2*length);
56     }
57
58     /** Returns a self-descriptive string of this instance in the form of
59     Rectangle(Shape=Circle, filled=, width=, length=) */
60     public String toString() {
61         return "A Rectangle with width = " + getWidth() + " and length = "
62             + getLength() + ", which is a subclass of " + super.toString();
63     }

```

Square.java

```

11 public class Square extends Rectangle {
12     // Constructor
13     public Square() {
14         super();
15     }
16
17     public Square(double side) {
18         super(side, side);
19     }
20
21     public Square(double side, String color, boolean filled) {
22         super(side, side, color, filled);
23     }
24
25     // Getter and Setter
26     public double getSide() {
27         return super.getWidth();
28     }
29
30     public void setSide(double side) {
31         super.setWidth(side);
32     }
33
34     public void getWidth(double side) {
35         super.setWidth(side);
36     }
37
38     public void getLength(double side) {
39         super.setLength(side);
40     }
41
42     /** Return a self-descriptive string of this instance in the form of
43     Square(Shape=Circle, filled=, side=) */
44     public String toString() {
45         return "A Square with side = " + getSide() + ", which is a subclass of "
46             + super.toString();
47     }
48 }

```

TestShape.java

```
1 public class TestShape {
2     public static void main(String[] args) {
3         // Declare and allocate a new instance of Shape
4         // with default values and filled
5         Shape s1 = new Shape();
6         System.out.println(s1.toString());
7
8         // Declare and allocate a new instance of Shape
9         // with name, color, and filled
10        Shape s2 = new Shape("red", false);
11        System.out.println(s2.toString());
12
13        // Declare and allocate a new instance of Circle
14        // with name, radius, color, and not filled
15        Circle c1 = new Circle(7.0, "red", false);
16        System.out.println(c1.toString());
17        System.out.println("luas = " + c1.getArea());
18        System.out.println("keliling = " + c1.getPerimeter() + "\n");
19
20        // Declare and allocate a new instance of Circle
21        // with name, width, height, color, and not filled
22        Rectangle r1 = new Rectangle(3.0, 5.0, "blue", true);
23        System.out.println(r1.toString());
24        System.out.println("luas = " + r1.getArea());
25        System.out.println("keliling = " + r1.getPerimeter() + "\n");
26
27        // Declare and allocate a new instance of Circle
28        // with name, width, height, color, and not filled
29        Square s4 = new Square(3.0, "yellow", true);
30        System.out.println(s4.toString());
31        System.out.println("luas = " + s4.getArea());
32        System.out.println("keliling = " + s4.getPerimeter());
33    }
34}
```

3. Masalah yang Dihadapi

- Tulis dua subclass dari shape yang disebut circle dan rectangle
- Tulis kelas square sebagai subclass dari rectangle
- Square dapat dimodelkan sebagai subclass dari rectangle. Kotak tidak memiliki instance variabel, namun mewarisi lebar dan panjang variabel dari superclassnya Rectangle

4. Solusi dari Masalah yang Dihadapi

- Kasus ini tidak berbeda jauh dengan task 1 yang memerlukan 3 konstruktor dan menggunakan setter dan getter

5. Teman yang Membantu

- Tidak ada

Task 3.1 – 3.2

1. Hasil Program

```
LIST :
Antonio Rossi 2100000.0 1989
Maria Bianchi 3012500.0 1991
Isabel Vidal 3150000.0 1993
SORTED LIST :
Isabel Vidal 3307500.0 1993
Maria Bianchi 3630062.5 1991
Antonio Rossi 2205000.0 1989
BUILD SUCCESSFUL (total time: 0 seconds)
```

2. Jawaban dari Soal yang Dipertanyakan

Employee.java

```
class Employee extends Sortable {
    // Instance Variables
    private String name;
    private double salary;
    private int hireYear;
    private int hireMonth;
    private int hireDay;

    public Employee(String n, double s, int day, int month, int year) {
        name = n;
        salary = s;
        hireDay = day;
        hireMonth = month;
        hireYear = year;
    }

    public void print() {
        System.out.println(name + " " + getSalary() + " " + hireYear());
    }

    public void raiseSalary(double byPercent) {
        salary *= 1 + byPercent / 100;
    }

    public int hireYear() {
        return hireYear;
    }
}
```

Manager.java

```
import java.util.*;

class Manager extends Employee {
    private String secretaryName;

    public Manager(String n, double s, int d, int m, int y) {
        super(n, s, d, m, y);
        secretaryName = "";
    }

    public void raiseSalary(double byPercent) {
        // add 1/21 bonus for every year of service
        GregorianCalendar todayDate = new GregorianCalendar();
        int currentYear = todayDate.get(Calendar.YEAR);
        double bonus = 0.5 * (currentYear - hireYear());
        super.raiseSalary(byPercent + bonus);
    }

    public String getSecretaryName() {
        return secretaryName;
    }

    public int compare(Sortable b) {
        Employee eb = (Employee) b;
        if (getSalary() < eb.getSalary()) return -1;
        if (getSalary() > eb.getSalary()) return +1;
        return 0;
    }
}
```

ManagerTest.java

```
public class ManagerTest {  
    public static void main(String[] args) {  
        Employee[] staff = new Employee[3];  
        staff[0] = new Employee("Antonio Rossi", 2000000, 1, 10, 1989);  
        staff[1] = new Manager("Maria Bianchi", 2500000, 1, 12, 1991);  
        staff[2] = new Employee("Isabel Vidal", 3000000, 1, 11, 1993);  
        System.out.println("LIST : ");  
        int i;  
        for (i = 0; i < 3; i++) staff[i].raiseSalary(5);  
        for (i = 0; i < 3; i++) staff[i].print();  
        Sortable.shell_sort(staff);  
        System.out.println("SORTED LIST : ");  
        for (i = 0; i < 3; i++) staff[i].raiseSalary(5);  
        for (i = 0; i < 3; i++) staff[i].print();  
    }  
}
```

Employee.java

```
public class EmployeeTest {  
    public static void main(String[] args) {  
        Employee[] staff = new Employee[3];  
        staff[0] = new Employee("Antonio Rossi", 2000000, 1, 10, 1989);  
        staff[1] = new Employee("Maria Bianchi", 2500000, 1, 12, 1991);  
        staff[2] = new Employee("Isabel Vidal", 3000000, 1, 11, 1993);  
        System.out.println("LIST : ");  
        int i;  
        for (i = 0; i < 3; i++) staff[i].raiseSalary(5);  
        for (i = 0; i < 3; i++) staff[i].print();  
        Sortable.shell_sort(staff);  
        System.out.println("SORTED LIST : ");  
        for (i = 0; i < 3; i++) staff[i].raiseSalary(5);  
        for (i = 0; i < 3; i++) staff[i].print();  
    }  
}
```

Sortable.java

```
abstract class Sortable {  
    public abstract int compare(Sortable o);  
    public static void shell_sort(Sortable[] a){  
        //shell sort body  
        int n = a.length;  
  
        // start with a big gap, then reduce the gap  
        for (int gap = n/2; gap > 0; gap /= 2)  
        {  
            for (int i = gap; i < n; i += 1)  
            {  
                // shift all the elements that have been gap  
                // sorted past a[i] in temp and make a hole at  
                // position i  
                Sortable temp = a[i];  
  
                // shift smaller gap-sorted elements up until  
                // the correct location for a[i] is found  
                int j;  
                for (j = i; j >= gap && a[j - gap].compare(temp) < 0; j -= gap)  
                    a[j] = a[j - gap];  
  
                // put temp (the original a[i]) in its correct  
                // location  
                a[j] = temp;  
            }  
        }  
    }  
}
```

3. Masalah yang Dihadapi
 - Tidak mengetahui dan baru pertama kali mendengar shortshell
4. Solusi dari Masalah yang Dihadapi
 - Mencari arti dan memahami konsep serta memodifikasi sesuai dengan kasus ini
5. Teman yang Membantu
 - Tidak ada